Master in High Performance Computing

Toward the Forecasting of Volcanic Plumes

Supervisors:
Tomaso Esposti Ongaro
Matteo Cerminara

Affiliations:
INGV - Sezione di Pisa
OGS - Trieste

Candidate:
Stella Valentina Paronuzzi Ticco

2nd edition
2015–2016

Contents

Chapter 1

Introduction

Figure 1.1: Calbuco eruption seen from Puerto Montt (Chile), about 20 miles southeast of the crater. This city is the main sea port at the lower end of Chile's western continental land, and is home to nearly 220.000 people.

On April the 22th of 2015, Calbuco volcano has erupted (Fig. 1.1) for the first time in 42 years, billowing a huge ash cloud over a sparsely populated, mountainous area in southern Chile. Local Airlines company had to cancel flights to and from neighboring cities, due to the presence of volcanic ash, which can potentially damage aircraft and make flying dangerous. The government immediately started distribute water in case resources were contaminated by ash; police and military officers were readily been deployed to ensure safety and help with evacuations; medical forces were sent because volcanic ashes may create serious respiratory diseases and illness. Naturally this was not an isolated event during 2015, nor this kind of emergencies are rare: out of an estimated 1.500 active volcanoes worldwide, 50 or so erupt every year, but not all of this natural catastrophe happens to be at actively monitored sites. As the regional emergency director of the Los Lagos region (Chile) said about

the above mentioned eruption: "For us it was a surprise, Calbuco was not under any special form of observation". Is clear then the necessity to develop early warning tools able to quantitatively forecast (for a period of several hours up to a few days) the dynamics of these processes in order to design appropriate emergency plans and mitigation measures.

It was for this urge of monitoring that firstly in 2008 Barsotti, et al. [1] (INGV, Sezione di Pisa) proposed a web-based warning tool for the modeling and forecasting ash loading and dispersal in the atmosphere (MAFALDA). By combing a simplified, one-dimensional model of the volcanic plume, together with weather forecasting data and a three dimensional, Lagrangian, advection-diffusion model, this application produces 2-D maps of airborne ash distribution at different altitudes and of ash deposits on the ground. Thanks to the fast execution time of the whole routine, MAFALDA can timely provide information on multiple pre-defined scenarios of an expected event. The entire procedure is routinely executed and since 2006 has been used it to produce ash forecast maps over the Etna region.

In 2009 another similar warning tool, was developed by Scollo et al. [2] (INGV, Sezione di Catania) with the aim of monitor volcanic activity near the airports (two) of the city of Catania: again whether forecast data are used, this time along with an empirical correlation to predict the height of tephra release [1] and a simplified atmospheric dispersal model to produce hazard maps of volcanic ash dispersal and deposition for predefined *scenarios*. Simulations are based on eruptive scenarios obtained by analyzing field data collected after the end of early Etna eruptions. Forecasting is supported by plume observations carried out by a monitoring system based infrared measurements, visual and thermal cameras able to detect ash dispersal and fallout. This tool has been used daily by the civil protection from 2007 on, to give advise on changing landing plans of airplanes in Catania airports, in case of small to mid-sized eruptions.

On this background I analyze in this work a new forecasting tool, based on ASHEE [3], a code developed at INGV, Sezione di Pisa, by M. Cerminara and T. Esposti Ongaro. The interesting new feature of this appliance, when fully operative, will be the capability of retaining a relatively small computational efforts while carrying out full 3D simulation of volcanic plumes. With respect to the models used in previous applications, ASHEE specifically addresses the fluid dynamic of volcanic plumes solving the full Eulerian transient mass, momentum, and energy equations for the plume mixture and ambient air in a three dimensional atmospheric domain. With respect to one-dimensional integral models, a 3D approach can describe the non-homogeneous features of a volcanic plume, i.e., the time and space dependent distribution of the concentration, temperature, pressure, and velocity of each constituent of the eruptive mixture, and the multiphase flow features of the eruptive mixture. In addition, turbulent flow can be explicitly simulated by resolving the eddy structure of the plume and the stratification and flow circulation without the need of calibrated, empirical parameters. Although 3D models were developed for volcanological applications in the 1990s, only in the last decade 3D simulations have become computationally affordable thanks to the advent of high-performance

[1] Tephra is one of the main products of explosive eruptions after material has been explosively ejected from a vent producing an eruption column.

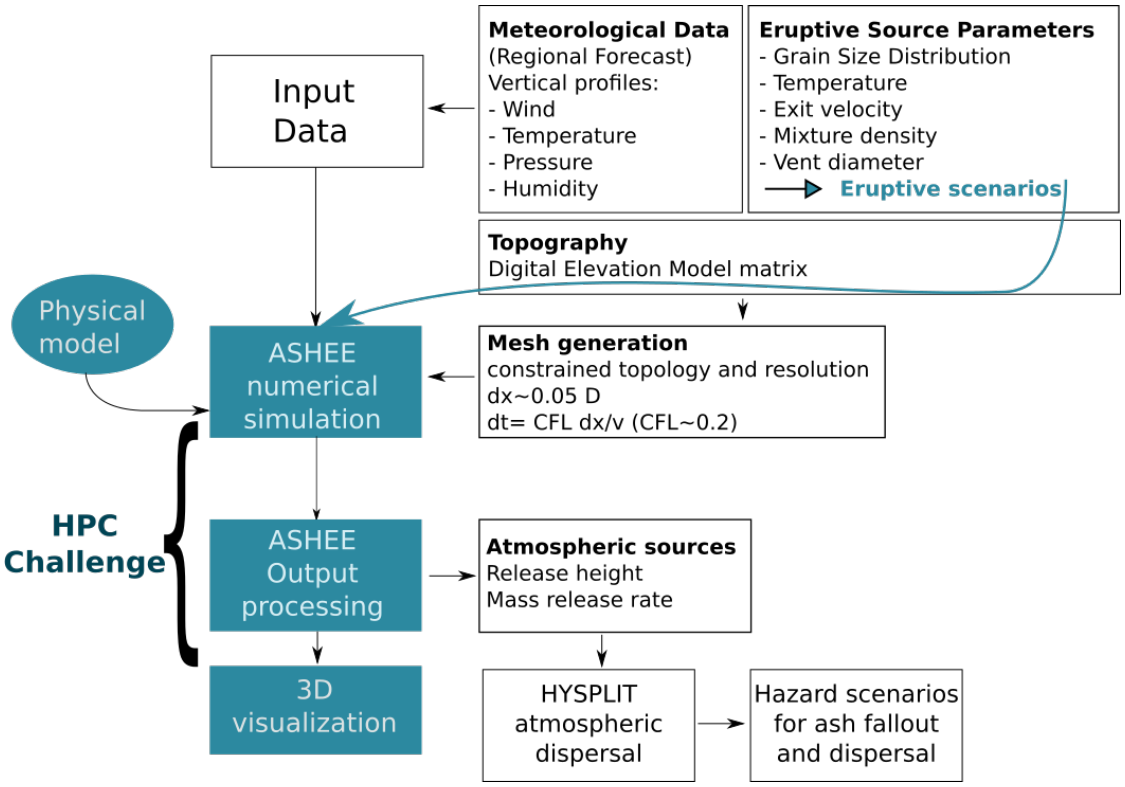

Figure 1.2: .

computing.

1.1 Volcanic Plume Forecasting Workflow

Figure 1.2 represents the typical workflow implementing the volcanic plume forecasting procedure. Starting from meteorological data, supplied by regional forecast centers, vertical profiles for wind, temperature, pressure and humidity need to be extracted at the volcano location. Additionally a digital elevation model can be used to take into account the topography surrounding the volcano vent. Eruptive source parameters (ESP) are then set for the target volcano and for a number N of pre-selected "eruptive scenarios". These are: grain size distribution, temperature, exit velocity and mixture density of the magma plus the vent volcano diameter. For each scenario, the plume model predicts the height and release rate of ash in the atmosphere (depending on both ESP and wind conditions). An atmospheric dispersal tool, such as HYSPLIT [6], can be chained to this forecasting process, to produce an hazard map for ashes fallout and dispersal, based on data produced by the simulations post-processing.

1.2 The Need for High Performance Computing

As already briefly outlined, the aim is the use of a three-dimensional model as a simulation tool for volcanic plumes. The need is thus to run contemporary

N scenarios and in real time extract the outputs to be able to start running the atmospheric dispersal tool in the smaller possible time. Not only for this reason we are tackling here a real HPC challenge. Let's write some numbers: to simulate a Plinian eruption, to get reasonable results we will need at list $\simeq 10^6$ cells for the mesh; to get interesting insight on the physic of the plume we need $\simeq 10$ time this number. On Fermi[7] super computer (CINECA infrastructure), this simulation lasts *5 days on 1024 cores*: this is clearly unacceptable, even admitting that Fermi is not the best possible choice for this kind of simulations. But this is not the only problem: since we need to run N of such simulation we will have to analize and postprocess a **big amount of data**. Estimating every cell to occupy 1-2 KiB, and hypothesizing around 200 saving points, this sums up to ~10 TiB of data for a single scenario, at resolution $24 * 10^6$ cells.

For the reasons here explained I considered interesting to measure ASHEE parallel performance, and possible strategies to improve this simulations efficiency. In the following work, I start presenting in Chapter 2, a brief review of the ASHEE model and computational environment, to subsequently introduce, in Chapter 3 the study, that I have made, on its parallel performance. In Chapter 4 I will describe how a parallel Lagrangian library has been integrated in the ASHEE code, to improve the physical description of volcanic eruptions. Therein I compare the performance of this enhanced code to the old one. In the end I will make a short summary on the obtained results, and a brief discussion is presented on the work that still has to be done, to effectively enable this tool for the active monitoring at sites of interest.

Chapter 2

ASHEE

In 2016 M. Cerminara, T. Esposti Ongaro and L. Berselli presented a new computational work [3] in which turbulent gas-particle forced plumes in the atmosphere were studied. In this paper a new fluid-dynamic model along with a C++ code was presented, the chosen name for this being ASHEE: ASH Equilibrium Eulerian. Theoretical and computational approaches to this thematic have been different over the years: models currently used in operational forecasting, solve the so called mono-dimensional pseudo gas or "dusty gas" model, in which both thermal and kinematic equilibrium are assumed between gas and particles forming the plume, and cylindrical symmetry (with respect to a generalized curvilinear axis) is assumed. Such models assume self-similarity and depend strongly on an empirical entrainment coefficient to describe the plume turbulent motion. To try and overcome limitations of these models, 3D steady-state dusty-gas codes have been developed: for the first time in 2010 [4] the entrainment coefficient was theoretically determined[1]. Steady-state dusty-gas models of volcanic plumes have thus had a formidable role in volcanology to identify the main processes controlling their dynamics and scaling properties, but are limited by the equilibrium assumption. To reproduce *non-equilibrium* dynamic, then, Eulerian multiphase flow models were developed (see, for example Ref. [9]) where $4 + I + 5J$ partial differential equations for the balance of mass, momentum and energy were solved, in presence of I gas components, and J particles type. This problem is clearly extremely heavy by a computational point of view, and this has so far hindered a detailed study of turbulent properties in multiphase flows.

The 3D ASHEE code implements a new physical model, able to resolve temporal and spatial interaction scales between gas and particles in turbulent regimes, and to describe the kinetic non-equilibrium dynamics and its influence on the *observable* features of volcanic plumes, while retaining thermal equilibrium. This model is a generalization of the dusty-gas model, keeping the kinematic non-equilibrium as a first-order correction, that generalizes Ferry and Balachandar's work of 2001 [10] to the compressible case. Although up to now ASHEE has been used to study subsonic plumes regimes, this model is also suited to work in transonic[2] and supersonic

[1] In one-dimensional, time-averaged models, the entrainment coefficient relates the influx of atmospheric air to the local, vertical plume velocity. The entrainment coefficient also determines the plume shape.

[2] It refers to the phase in which the eruptive mixture is injected into the atmosphere: the

regimes.

2.1 The Equations

The ASHEE model is conceived for resolving dilute suspensions, namely mixtures of gases and particles with volumetric concentration $\frac{V_s}{V} \equiv \epsilon_s \lesssim 10^{-3}$, corresponding to regimes in which particle-particle collisions can be disregarded. In volcanic plumes this threshold is usually overcome only near the vent, i.e. in a region small when compared to the entire plume extension. Moreover this formulation is supposed to describe situation in which particle Stokes number [3] does not exceed about 0.2. I will address cases in which these two threshold are overcome in Sec. 4. For the time being I will suppose these conditions to hold. In this case, the multiphase flow model reduces to a Navier-Stokes model for a gas-particle mixture, with a modified equation of state, additional mass transport equations for gas and particles species, and some extra-terms accounting for momentum non-equilibrium. In contrast, the mass fraction of the solid particles cannot be considered small, because particles are heavy: particle inertia must be taken into account considering a two-way coupling between gas and particles. The Equilibrium-Eulerian model in mixture formulation thus reads:

$$\partial_t \rho_m + \nabla \cdot (\rho_m \boldsymbol{u}_m) = 0 \tag{2.1a}$$

$$\partial_t (\rho_m y_i) + \nabla \cdot (\rho_m \boldsymbol{u}_g y_i) = 0, \quad i \in I \tag{2.1b}$$

$$\partial_t (\rho_m y_j) + \nabla \cdot (\rho_m \boldsymbol{u}_j y_j) = 0, \quad j \in J \tag{2.1c}$$

$$\partial_t (\rho_m \boldsymbol{u}_m) + \nabla \cdot (\rho_m \boldsymbol{u}_m \otimes \boldsymbol{u}_m + \rho_m \mathbb{T}) = -\nabla P + \nabla \cdot \mathbb{T}_r + \rho_m \boldsymbol{g} \tag{2.1d}$$

$$\begin{aligned} \partial_t (\rho_m \boldsymbol{h}_m) + \nabla \cdot [\rho_m h_m (\boldsymbol{u}_m + \boldsymbol{v}_h)] = &+ \partial_t p - \partial_t (\rho_m K_m) \\ &- \nabla \cdot [\rho_m K_m (\boldsymbol{u}_m \boldsymbol{v}_K)] \\ &+ \nabla \cdot (\mathbb{T} \cdot \boldsymbol{u}_g - \boldsymbol{q}) + \rho_m (\boldsymbol{g} \cdot \boldsymbol{u}_m) \end{aligned} \tag{2.1e}$$

where $y_i(x,t)$ and $y_j(x,t)$ are the mass fraction for the I gaseous and J solid phases respectively; $\rho_m(x,t) = \sum_i \rho_i + \sum_j \rho_j$ is the density of the mixture; $\boldsymbol{u}_m(x,t)$ and $\boldsymbol{u}_j(x,t)$ are the velocity fields; $h_m(x,t)$ is the enthalpy; the terms $K_g = \frac{1}{2}|\boldsymbol{u}_g|^2$ and $K_j = \frac{1}{2}|\boldsymbol{u}_j|^2$ are the kinetic energy per unity of mass of the gaseous and solid phases, respectively. Here all the source (or sink) term are assumed to be zero for sake of clarity. The first equation is redundant, because it is contained in the second and third set of continuity equations and $\sum_I y_i + \sum_J y_j = 1$. Hence the main differences with respect to a pure Navier-Stokes formulation are: the tensor term $\rho_m \mathbb{T}$ in Eq. (2.1d) and the terms for v_h and v_k in Eq. (2.1e), that account for differences in velocity between gas and particle mixture (also implying a difference in kinetic energy). Eq. (2.1b) and (2.1c) are the continuity equation for the solid and the gaseous phases. The correction to particle velocity up to first order is obtained by using the Stokes law and a perturbation method on the Lagrangian particle momentum balance, and reads:

$$\boldsymbol{u}_j = \boldsymbol{u}_g + \boldsymbol{w}_j - \tau_j (\partial_t \boldsymbol{u}_g + \boldsymbol{u}_j \cdot \nabla \boldsymbol{u}_g) + O(\tau_j^2). \tag{2.2}$$

pressure can be higher than P_{atmo} in which case the flow is initially driven by a rapid, transonic decompression stage.

[3] St - i.e., the ratio between particles relaxation time and flow characteristic time.

where τ_j is the characteristic time of particle velocity relaxation with respect the gas. The differences just highlighted bring to a computational algorithm that retains more or less the same convergence properties of a standard Navier-Stokes implementation. The real important difference reside in Eq.(2.1b) and (2.1c) that have to be solved for *all the phases* in the mixture: this clearly brings an increase in computational load with respect to the standard case.

I will skip here other technical details about the mathematics lying behind ASHEE model, because this goes far beyond the goal of the present work. The interested reader can find the whole derivation excellently explained in Ref. [3].

2.2 The OpenFOAM Platform

The ASHEE code solves numerically the Eulerian model described in Sect. 2.1 to obtain a time-dependent description of all independent flow fields in a three-dimensional domain with prescribed initial and boundary conditions. The developers has chosen to adopt an open-source approach to the code in order to guarantee control on the numerical solution procedure and to share scientific knowledge. As a platform for developing the solver, they chose the unstructured, finite volume (FV) method, open-source C++ library, OpenFOAM ®: it is released under the Gnu Public License (GPL) and has gained a vast popularity in recent years. The already existing solvers and tutorials provide a quick start to using the code also to rather inexperienced users. On the other hand the exasperated use of templated classes tends to generate confusion if one has to integrate provided libraries in to a new solver (e.g., to solve a different set of equations) and/or to implement new numerical schemes. Anyway the integration of advanced tools for pre-processing (including meshing) and post-processing (including visualization) makes OpenFOAM an undeniably excellent platform. Finally, all solvers can be run in parallel on distributed memory architectures, which makes OpenFOAM suited for envisaging large-scale, three-dimensional volcanological problems. The new computational model, called ASHEE is documented in the VMSG [14] (Volcano Modeling and Simulation Gateway) at Istituto Nazionale di Geofisica e Vulcanologia and is made available through the VHub [15] portal.

Chapter 3

Parallel Performance

In this Chapter, as first thing, a description of the way in which the simulation scenario influences the computational aspect is presented. Subsequently a study on ASHEE parallel performance is made. This study determines a sort of prescription for running this kind of simulations in the most efficient possible way. A look inside the openMPI parallelization is taken by means of the IPM monitoring tool, trying to give an explanation to the scaling behavior encountered during performance study.

3.1 Numerical Simulation Parameters

As already introduced talking about the work-flow diagram, the actual parameters that determine a volcanic event are the eruptive source parameters (ESP). These are important also from a computational point of view, determining both the resolution required to simulate a given scenario, and the time that this will take. Let's distinguish, from now on, between *strong plumes* and *weak plumes*. The term weak plume will indicates all those eruptions, (e.g. Shinmoe-dake, Japan 2011), characterized by a $\phi_{\text{Mass}} < 10^7$ Kg/s and a volcano vent diameter smaller than about 50 meters. With the term *strong plume* (e.g. Pinatubo, Philippine, 1991) will be characterized the eruptions with a $\phi_{\text{Mass}} > 10^8$ Kg/s and a volcano vent diameter larger than about 0.5 Km. For a computational model the minimum requirement to consider a simulation reliable is its consistency. Consistent means that the average values of important properties will stay constant at variance with the resolution. In Ref. [5] authors make a detailed study on the condition under which this constraint is satisfied for the ASHEE code. Specifically, they found that: in the case of a weak plume, in going from medium to high resolution, the average values have acceptably small variation, making a medium resolution good enough to study this kind of scenarios[1]. For a strong plume scenario, on the other hand, average properties variation are not negligible zooming from medium to high resolution, making unavoidable the use of a high resolution mesh for this kind of simulations. This is due to the fact that, in this latter case, the plume tend to collapse near the vent, making hard the description of the entrainment, and as a consequence, of the overall physical properties. Since the definition of "high",

[1] This is not true in going from low to medium resolution, though.

CPU	Intel(R) Xeon(R) CPU E5-2630 v3
Speed	2.40 GHz
OS	CentOS release 6.6
OpenMPI	openmpi-1.6.5/openmpi-1.6.2
C compiler	mpicc
Link layer	InfiniBand QDR IB (40Gb/s)

Table 3.1: Specs of the hardware and software equipment used for the tests.

"medium" or "low" resolution is rather peculiar of the context, I'll give an estimate of what this means within this work:

1. Low resolution: 8 cells in the volcano diameter. $N_{\text{cells}} \lesssim 10^5$.
2. Medium resolution: more than 8 and less than 16 cells in volcano vent diameter. $10^5 \lesssim N_{\text{cells}} \lesssim 10^6$.
3. High resolution: 16 or more cells in volcano vent diameter. $N_{\text{cells}} \gtrsim 10^7$.

Now: a difference in mesh sizes will imply also a difference in simulation times, and this is due to the constraint imposed by the Courant-Friedrichs-Lewy condition:

$$\frac{u * \Delta t}{\Delta x} = C_o$$

Since C_o has to be ≤ 0.2 for convergence reasons, this means that a decrease in Δx will imply a decrease in Δt, increasing the total simulation time. Summarizing, even though roughly the same number of cells can be used in both cases[2], strong plumes simulations are more computationally expensive because they require a smaller time steps.

3.2 The Measurements

Has already stated in introducing this work: the goal here is to enable the use of ASHEE as a forecasting tool. This means that there will be a target time, into which results produced by outputs post-processing has to be ready. This can be either 12 or 24 h, depending on the broadcast of new meteorological data: ensuring the simulations are done in due time implies that a thorough knowledge of execution times and scaling properties of the application is needed. For this reason I present here a study on *ASHEE parallel performance*. Since the main part of the work has been carried out at INGV-Pi, most of the tests has been run on Laki cluster, that is a brand newly installed machine (March 2016), composed of standard high-volume servers, a Gigabit Ethernet network and a high-performance

[2]Here plays a role also the fact that the mesh is non-homogeneous, and difference between δx_{min} and δx_{max} can become considerable.

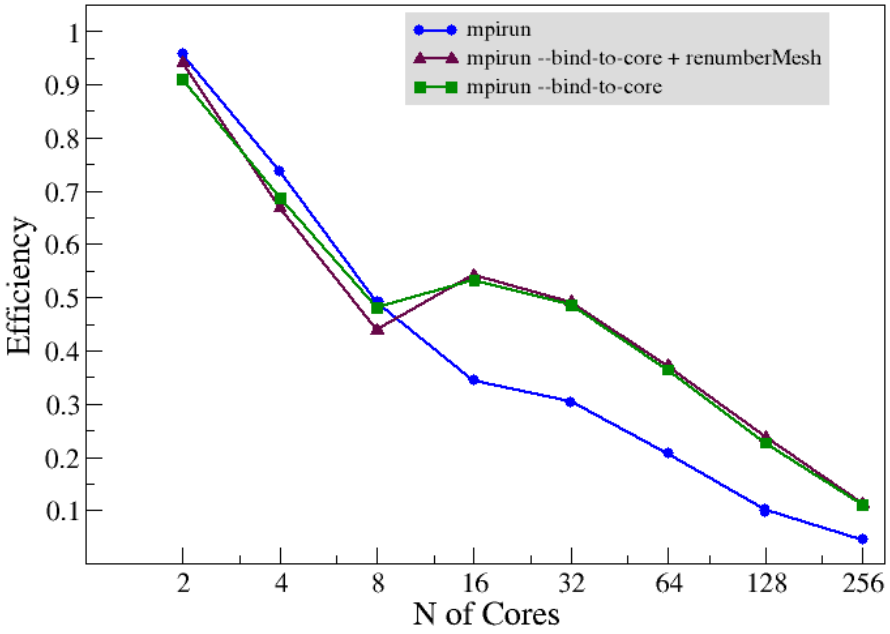

Figure 3.1: Strong scaling for low resolution mesh: combining different Open MPI options with OpenFOAM we can gain a lot in performance: -bind-to-core option is of crucial importance when running on great N of cores. Another small gain can be achieved if the renumberMesh routine is used, in fact the purple line is systematically higher than the green one. x-log scale is applied.

InfiniBand interconnection. The cluster is made by 4 chassis containing 4 nodes + 1 Master Node, each node having 2 Haswell cpus[3], and 64 GB of RAM. All 256 cores where at disposal for this study. These and others hardware specifications are summarized in Tab. 3.1. First thing to do is then: run the application and try to understand if there are regime in which its behavior (from a computational point of view) is better than in others. For a parallel application this means a study on scaling properties and a study on the work load balance over N processors, when N is varying. For the preliminary tests have been chosen small plume scenario with very low resolution. I will use the term *low* resolution for simulation with a mesh usually less than about 10^5 cells (less than 8 cells in the volcano diameter).

I performed a *strong scaling* measure, that consists in running the same simulation on an increasing number of cores. In doing so, I studied different combination of OpenMPI option with OpenFOAM specific routines for mesh renumbering. Results are reported in Fig. 3.1: comparing blue and purple line we have a clear indication that the `-bind-to-core` option for the `mpirun` is extremely beneficial. Anyway blue line is systematically higher from 1 to 16 cores, i.e. the part that we will refer to as *intra-node*. This is because without this option the operating system will move processes from core to core in an attempt to optimize channel memory usage. Such behavior is suited for half empty machines, but when at a full load tends to congest 3.2 the system leading to a situation in which a crush of the whole node

[3]With Haswell architecture this means two sockets with 8 cores each, summing up to 16 cores per node.

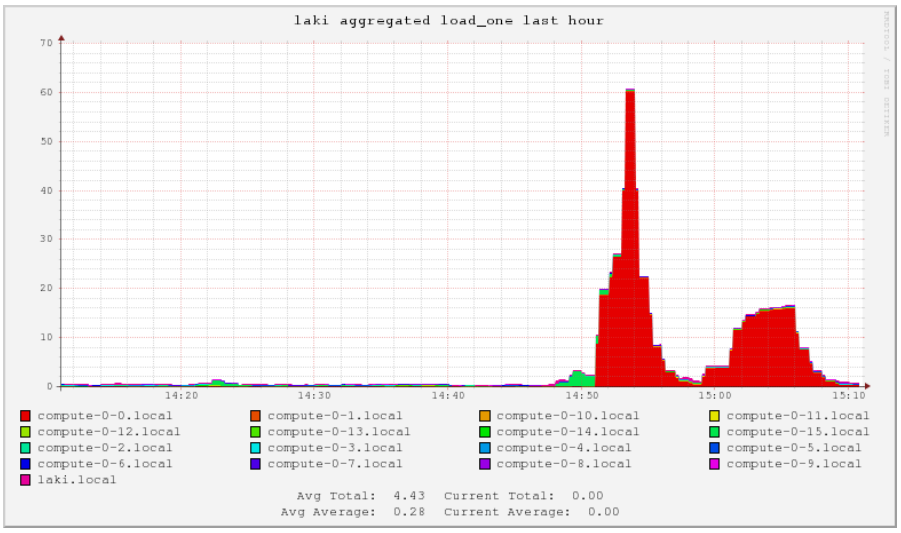

Figure 3.2: Two different runs: without and with the -bind-to-core MPI option. Running the same number of processes the total load in the first case (first peak) is roughly 4.5 times the load of the second (second peak). Screen taken from the Ganglia monitoring tool installed on Laki cluster.

is likely[4]. In addition, the `renumberMesh` utility of OpenFOAM has been tested: it renumbers the cell list in order to reduce the memory bandwidth and, as can be seen looking at the purple line, is useful when running on large numbers of cores. Summarizing: from now on all simulation I will talk about are run with the combination `-bind-to-core -renumberMesh`. This was of curse only a preliminary test, being the problem too small to study any scaling properties, and that is the reason of the small number reported on x axis in Fig. 3.1.

To make a proper analysis I use a strong plume case with two different resolutions, that I will call low and high respectively, with 12 and 20 cells in the volcano diameter. From Fig. 3.3 is clear that most of the performance is lost within the node and this affects the overall scaling properties. On the larger problem size (blue curve), the efficiency is about 100% when normalized to the CPU ($N = 8$) or node ($N = 16$) performance. For the lower resolution, the efficiency decreases due to the MPI communication overhead: 256 cores are too many for such a small problem, and the overhead added by the MPI infrastructure is much greater than the gain in resolution velocity. The high resolution mesh, on the other hand, is too memory-consuming to be run on less than 8 cores: the gain in performance increasing N up to 256 cores is clear. The optimal number of cores should therefore be tuned according to the problem size. To try and explain the intra-node behavior of the code, we have introduced specific instructions in the solver to report the timing of the main routines. They have been subdivided in four main categories: matrix assemblage; vector assemblage; matrix inversion; LES (Large Eddies Simulation) routines. They contribute all together to almost 100% of the total time of simulation. In Fig. 3.4 we report the results obtained on 4 cores for the high-resolution mesh:

[4]Usually, on large HPC infrastructure, like CINECA, the `-bind-to-core` option is implemented at OS level, making process binding the default. Probably process shuffling between cores is really useful only on platform, as laptops, intended for a completely different usage.

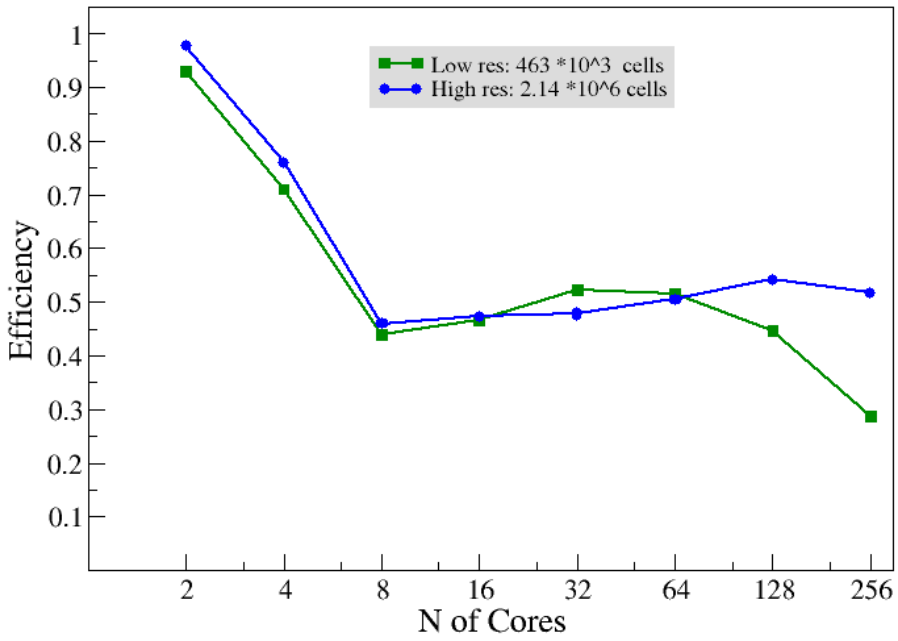

Figure 3.3: Strong scaling for low resolution mesh: combining different Open MPI options with OpenFOAM we can gain a lot in performance: -bind-to-core option is of crucial importance when running on great N of cores. Another small gain can be achieved if the renumberMesh routine is used, in fact the purple line is systematically higher than the green one. x-log scale is applied.

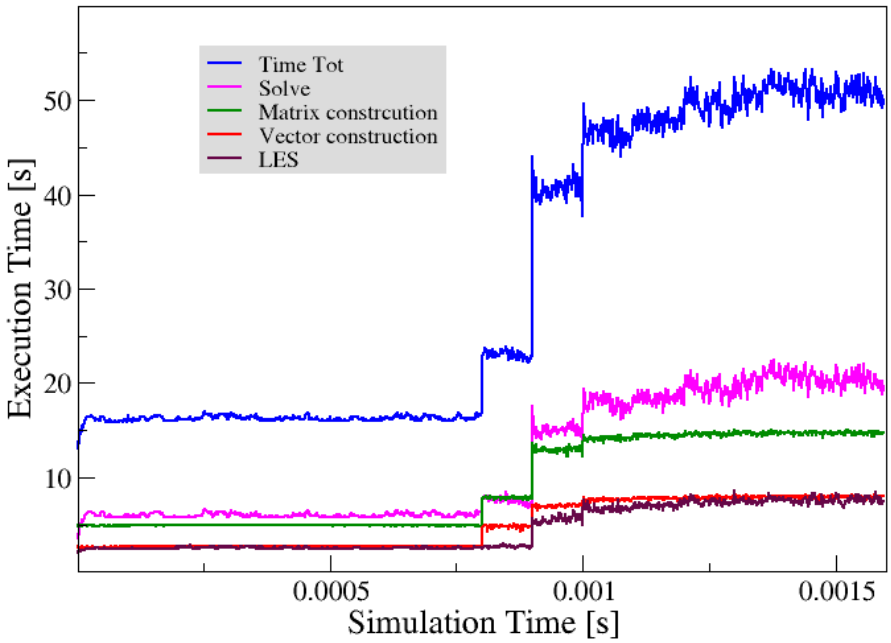

Figure 3.4: Manual timing on 4 cores for the high resolution mesh. The efficiency is near to 100 % for the first time steps, but then the total execution time start to increase in a sort of step-fashion, due to the increase of exec time of the routines that are being monitored all together.

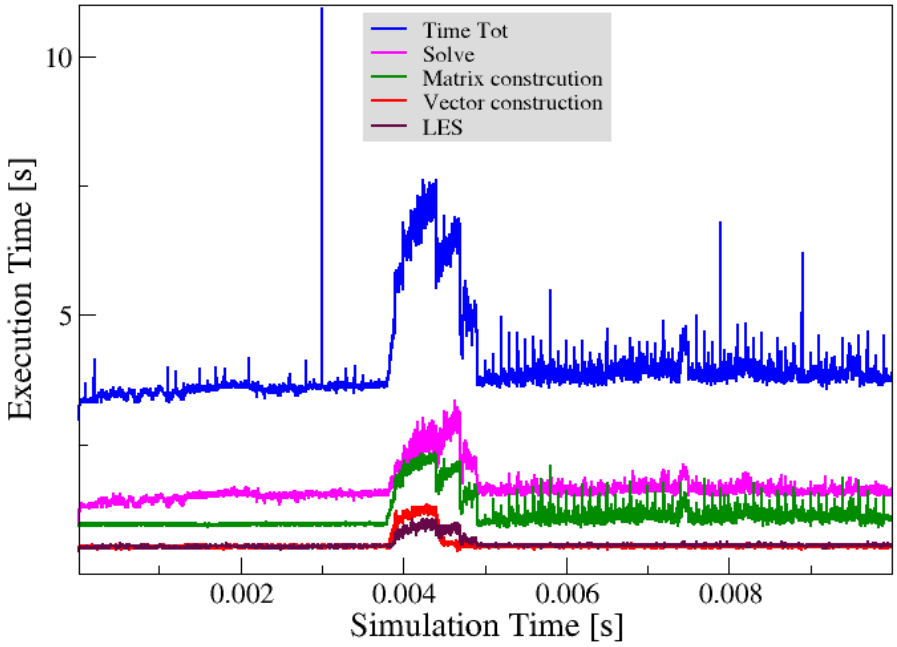

Figure 3.5: Manual timing on 32 cores for the high resolution mesh. The sort of step-structures disappear, replaced by a sort of hump, that slightly increase the overall time of execution. Again the bump involves all the routines and cannot be ascribed to a single functions.

in the first part of the simulation the scaling efficiency is near 100%. It then starts to increase at a random time-step several times. This increase cannot be ascribed to one single routine, but it is instead common to all the processes. This tests have been repeated, to see if some patterns can be highlighted, but the results have been discouraging: this jumps appear always, but at different time-steps, and also their height differs form run to run. However, looking at the same graph obtained on 32 cores 3.5 this jumps disappear, suggesting that the problem is node-bound.

3.3 The IPM Monitoring Tool

To have a better insight in possible underlying problematics a more professional tool is needed: I decided to go for the Integrated Performance Monitoring (IPM) tool [11]. This is a well known and portable profiling infrastructure that provides, at the cost of a really low overhead, insight on MPI performance aspects and resource utilization: communication, computation, and IO are the primary focus. The information is gathered in a way to minimize the impact on the running code, maintaining a small fixed memory footprint and using minimal amounts of CPU. The characteristic that, over all, makes IPM a tool worth using is that it allows monitoring without sources recompilation. A full detailed report, finally, can be obtained exploiting the possibility IPM offers to integrate the PAPI[5] monitoring tool [12]: although this is a great chance, on Haswell processors many of the

[5]The Performance API (PAPI) project specifies a standard application programming interface (API) for accessing hardware performance counters available on most modern microprocessors.

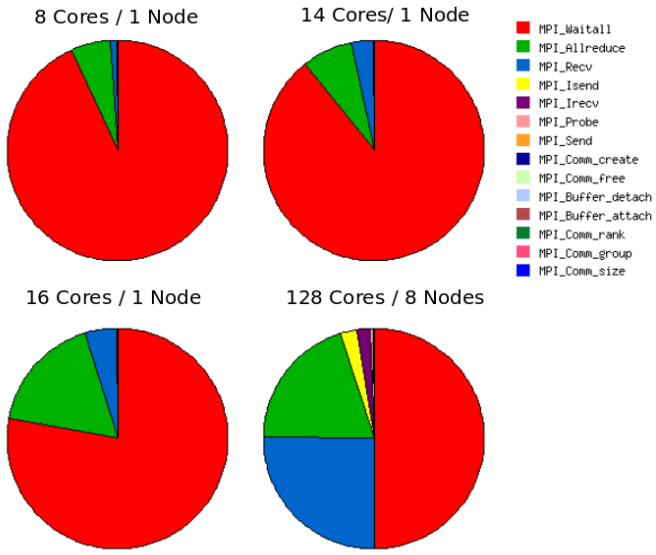

Figure 3.6: Distribution of mpi routines. Clearly when running on 8 cores little time is spent in send and receive data. Notice how increasing the number of nodes the `mpi Recv` rate increases. To have a proper understanding of this data anyway is important to have a look at Tab. 3.2 in which the absolute values are reported.

MPI Tasks	Wallklock [s]	`mpi` [%]	Tot Memory [GB]	cells/s/proc
8 on 1 Host	$8.6 * 10^3$	7.11	25.7	$12.2 * 10^3$
14 on 1 Host	$3.1 * 10^3$	17.27	26.5	$20.0 * 10^3$
16 on 1 Host	$2.5 * 10^3$	6.80	26.9	$20.9 * 10^3$
128 on 8 Host	$4.8 * 10^2$	9.75	36.1	$19.1 * 10^3$

Table 3.2: Absolute values of exec times, percentage of `mpi` communication with respect to tot time and total memory allocated for the case. This last has an increase of $\simeq 31\%$from 8 to 128 cores. This is a really important data to keep into account when thinking to scale to large number of cores.

hardware counter PAPI uses are disabled[13]. Not a second issue: since the use of PAPI implies direct intervention on the code, sources recompilation would be needed. A run of the code instrumented with IPM will produce an XML file, in which the data, collected by the profiler, are stored. IPM package include also a tool for processing data (the IPM parser) that will create human readable reports starting from the XML file.

For this set of tests I considered a strong plume with a high resolution mesh ($8.8 * 10^6$ cells). Fig. 3.6 and Tab. 3.2 report results obtained running an IPM instrumented version of ASHEE on 8, 14, 16 and 128 cores respectively. Taking a look at the pie charts it emerges that, over all the `mpi Allreduce` is the more time consuming call. For what concerns the rest of the MPI calls: as the number of cores increase the time spent in `mpi Allreduce` grows, and at a rate higher than the `mpi Recv`, whose percentage becomes important only when exiting the node. This data,

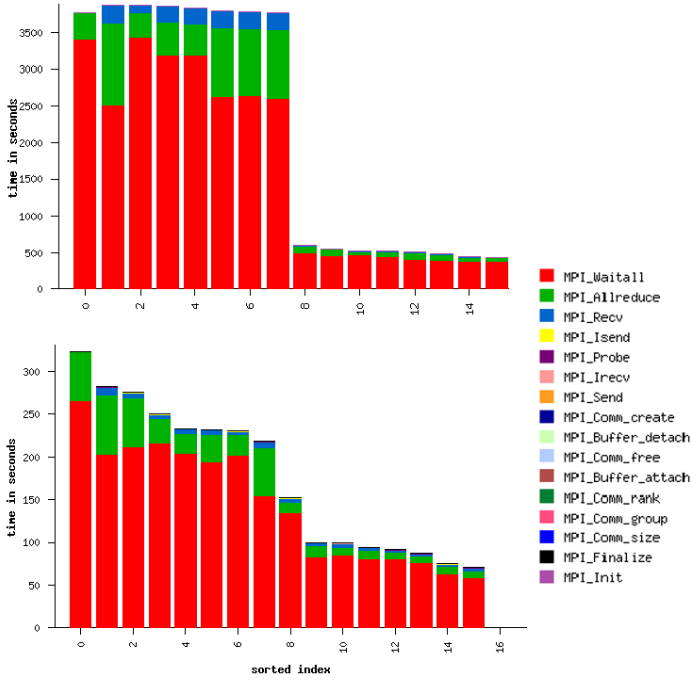

Figure 3.7: Comparison between cores load balance for two different run of the same scenario decomposed on 16 cores (for the lower: IPM data are reported in Tab. 3.2). The difference in the exection time (reported on y axis) is evident and concurrent with a great unbalance between the single cores load (here reported in descending order, not by rank).

anyway, are not significant until a comparison with absolute values of execution times is made: in Table 3.2 such data are reported. First thing to notice is the memory consumption: the allocated GBs increase by about 30% from 8 to 128 cores. This is to be taken into account when planning to run big simulations: the RAM needed cannot just be computed as 1-2 Kb * N_{cells}, because the decomposition of the mesh will increase this estimation, and by an amount that strongly depend on the number of cores. Taking now a look at the ratio between simulation time and time spent in `mpi` routines a sudden jump at 14 cores can be seen: while it is clear why percentage is slightly higher for 128 cores with respect to 16, [6] is not clear why it is so for $N = 14$. Furthermore: even if the final simulation velocity is rather good, I can say that this was, in facts, just a stroke of luck. I run a 16 core simulation (not reported here, but identical in all to that in Tab. 3.2) whose `mpi` time was 38% of the total time. In that case there was also a strong effect on the overall simulation performance, with a computing velocity of about $9.3 * 10^3$ cells/s/proc. I concluded that this is due to an occasional unbalancing between the cores load, as displayed in Fig. 3.7, where the exactly same test case is shown on two subsequent run. This can be ascribed either to some nasty trick played by the operating system, that is slowing down communication, or to some OpenFOAM parallelization issue that shows up only when a small number of cores is used for big problems like this. I think that this could also be the reason behind those jumps

[6] mpi has to communicate between 8 nodes, not only between cores, involving intervention of the Infiniband interconnection

detected in Fig. 3.2.

Summarizing, the real important informations emerged so far, are that: 1) to have a good scaling efficiency there is a minimum load per core, that is around 13.000 cell/proc 2) when N_{cores} is small with respect to N_{cells} the overall velocity of resolution can be hardly affected by cores load unbalance, causing the simulation time to become prohibitive. Finally: is to be kept in mind that the memory request for a defined scenario will not depend only on the mesh resolution, being the increase in GB due to decomposition not at all negligible.

Chapter 4

The Lagrangian Approach

Introducing the ASHEE model in chapter 2 I specified that: this model is intended for dilute regimes with a Stoke number St not exceeding 0.2. The Stokes number is a dimensionless parameter, used to characterize the behavior of particles suspended in a fluid flow. This is defined as the ratio between the particles relaxation time (i.e., the time needed to a particle to equilibrate to the gas velocity) and a characteristic fluid time scale, which for turbulent flows is related to the large-eddy average rotation time. Three different regimes can be delimited:

1. St $\leq 10^{-3}$. Dusty gas approach: gas and particles are almost perfectly *coupled*, meaning that, in term of the variables used in Sec. 2.1 $\boldsymbol{u}_i$ for a given particle is equal to $\boldsymbol{u}_g$. This description is suited, in volcanic plumes, for particles size up to about 50 μm (fine ash).

2. $10^{-3} \leq$ St ≤ 0.2. Equilibrium-Eulerian approach: particles are no longer perfectly coupled with the gas and move with a velocity $\boldsymbol{u}_j \neq \boldsymbol{u}_g$. Settling and preferential concentration phenomena start to appear and can be largely different from the one of the gas and the fine phase. This description can be used, in volcanic plumes, for particles up to 1 mm, i.e. for the "ash".

3. St ≥ 0.2. Lagrangian approach: the decoupling between gas and particles is almost complete, and especially in places where turbulence is strong the behavior of the gas and particles phases will be highly different. Both preferential concentration and settling phenomena can become really strong. With this approach parcel larger than 1 mm can be described, a thing that is obviously highly appealing since the aim is to describe a volcanic eruption, in which ballistic up to meters large can be expelled.

In this Chapter I will introduce the computational approach adopted to deal with this regime, and then examine the computational impact that this has on the ASHEE performance. Future developments are discussed.

4.1 The Lagrangian Library

The OpenFOAM platform offers mainly two options to deal with particles tracking: the Lagrangian `intermediate` library and the Lagrangian `solidParticle`

class. While the `solidParticle` library consists of a single class in which particles are tracked individually, the `intermediate` library consists of a series of classes, forces and objects, set up to track particles, or "computational parcel". The main difference between the `solidParticle` approach and the `intermediate` one is that the latter allows to describe parcels (i.e., clusters of particles) and to describe the two-way and the four-way coupling regimes. This means that the effects of the Lagrangian particles on the Eulerian phase and the interactions among particles can be modeled. In this work, however, I have just analyzed the one-way regime, because the mass flux of the injected Lagrangian particles is lower than 0.1% of the total, and their effect on the mixture can be safely considered small. This characteristic is also what makes this library suited for the ASHEE framework. More specifically, in the `intermediate` library more than one class is implemented, each suited to describe a different physical phenomena (chemical reaction, vaporization, etc.), many of which goes beyond the interest of this work. I choose to use the `KinematicCollidingCloud` class: this class tracks parcels, updating both their position and velocity, and gives the possibility to keep into account collisions between parcels. The integration of this within the ASHEE code implied a modification in the solved equations: Eq. 2.1d and Eq. 2.1e will now have extra terms accounting for the drag force. Among the drag models OpenFOAM implements I used

$$F_{\text{drag},i} = \frac{3}{4} \frac{m_i \, C_D(\text{Re}_\text{s}) \, \text{Re}_\text{s} \, \mu}{\rho_i \, d_i^2} \, |\boldsymbol{u}_g - \boldsymbol{u}_i|, \tag{4.1}$$

i.e. the Ergun-Wen-Yu drag model [16]. Here d and m are the particles diameter and mass respectively, while μ is the viscosity of the gas phase. $C_D(Re)$ is instead the drag coefficient, as a function of the particles Reynolds number, and reads:

$$C_D(\text{Re}_\text{s}) = \frac{24}{\text{Re}_\text{s}} \, (1 + 0.15 \text{Re}_\text{s}^{0.687}), \tag{4.2}$$

in accordance with Eq. 1 - Eq. 4 in Ref. [3]. Finally I use the `coneNozzleInjection`, that allows the user to select: start time of injection, injector position, direction and velocity (along injection axis) of the injected parcels, and their diameter.

4.1.1 ASHEE Lagrangian Performance

In this section I will refer, with the name ASHEE Lagrangian, to this new version of the code. As done in Chapter 3 for the ASHEE code, I will measure ASHEE Lagrangian performance, trying to compare the results to the previous case. For this measurement a strong plume scenario has been used, with a high resolution mesh, $N_{\text{cell}} = 8.8 * 10^6$, 32 cells in the vent diameter. The additional data needed here, to define properly the simulation scenario, are the parameters that characterize the Lagrangian phase: for this test 10^3 parcels/s have been injected into the atmosphere with a diameter of 8 mm, a density of 1300 Kg/m^3 and a mass of about 0.35 g. As it can be seen from Figure 4.1: the number of cells per second that the code is able to resolve diminishes increasing the number of Lagrangian parcels to track. To have a fair comparison between ASHEE Lagrangian and ASHEE, the approach has been to:

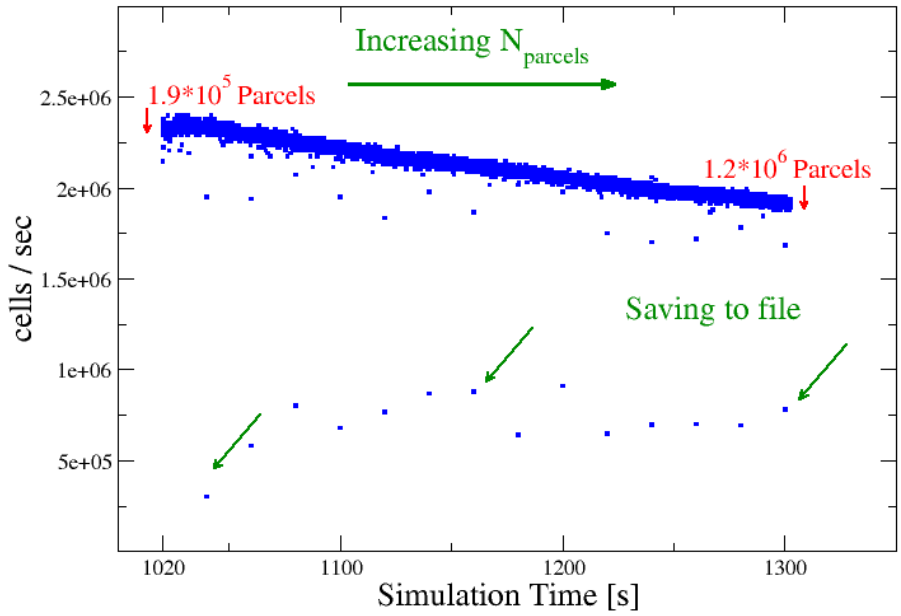

Figure 4.1: Speed of the simulation measured as cells/sec while injection is activated at a rate of 1000 parcel/sec. A growth in the number of Lagrangian parcels entail an increase in computational load, decreasing the overall speed. The time steps in which outputs files are written, obviously, the slowest.

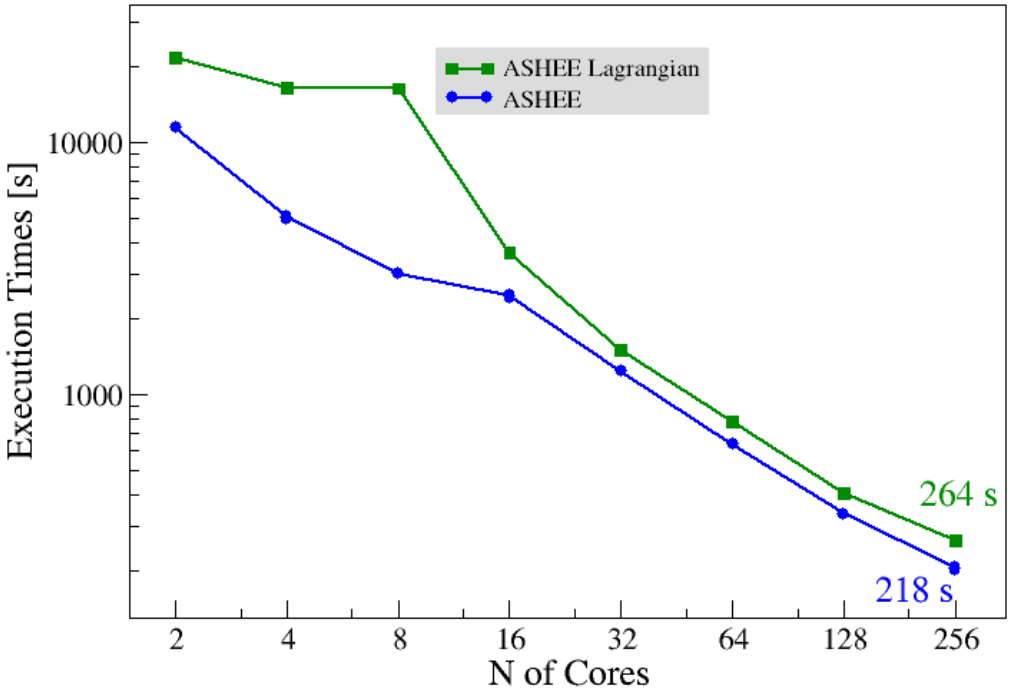

Figure 4.2: Timings for a 100 time-steps simulation at variance with N_{cores}. As could be expected the tracking of the Lagrangian phase increases the execution times. Exact values are specified for the 256 cores run. Logarithmic scale is applied both to x- and y-axis.

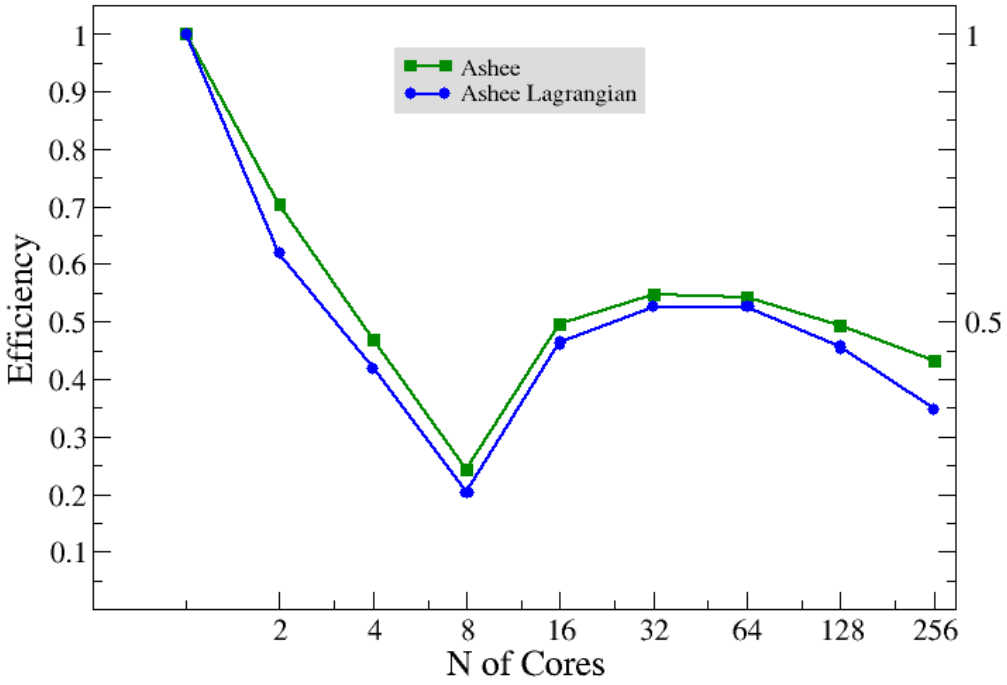

Figure 4.3: Strong scaling for high resolution mesh, comparing efficiency between ASHEE Lagrangian and ASHEE code. The intra-node behavior is quite dramatic for both, and the efficiency along the curve is systematically lower for the Lagrangian case, losing between 5% and 10% of efficiency. x-log scale is applied.

1. Run 1000 s of simulation with parcels injection activated.

2. Stop the injection, but keep on tracking the position and velocity changes of the Lagrangian phase. Run the simulation for 100 time steps, and take timings, varying N_{cores}.

3. Re-run the same 100 time steps, this time also shutting down Lagrangian parcels tracking.

With this procedure I obtained the data plotted in Fig. 4.2 and Fig. 4.3. Figure 4.2 reports the execution times for 100 time-steps, both when tracking is on and off. As expected, the tracking of Lagrangian parcels is increasing the execution time: at every time-step the code has now to compute and update both positions and velocities for all the parcels. Obviously this increase is also proportional to the number of tracked parcels. Since $N_{\text{p}} = 10^6$ is already a value that can be used in real simulation, I think that this data are rather significant, and give a good estimate of the impact that Lagrangian tracking has on the simulation times. For what concerns the scaling properties shown in Fig. 4.3, the first thing I would like to notice is that, with respect to Fig. 3.3, the intra-node part is more "loaded": a mesh of about $8.8 * 10^6$ cells with 8 cores at disposal means that each processor has to deal roughly with $1.1 * 10^6$ cells, a rather high value if compared to the prescription of about 15.000 cells/core obtained in Sect. 3.2. This partially explains the observed scaling behavior. Anyway what emerges is an overall loss of efficiency, of about 10%, that was not expected, since it means that not only Lagrangian

tracking is affecting the serial timings, but also that keep adding workload as N_{proc} increases. Since this can possibly be linked to the parallelization strategy adopted within the Lagrangian library, I tried to inquire this problem with the help of the IPM monitoring tool. Unfortunately, some problem emerges when running IPM along with the Lagrangian tracking activated: all the simulations run with ASHEE Lagrangian + IPM where using three time the walltime used without monitoring, when not aborting completely. In this case all the retrieved data has to be considered totally untrustworthy. Unluckily I had not enough time, during this thesis, to enter more deeply in this problematic. I think, however, that the investigation in this issue has to be addressed in the future, because it can possibly reveal what lies beneath the scaling behavior reported in Fig. 4.3 and can maybe suggest a way out. Anyway, for the time being this estimate can be made: for a mid resolution mesh with $N_{\text{cells}} \simeq 3 * 10^6$, and a total number of injected parcel of about $1 * 10^6$, the simulation of 1000 s using ASHEE Lagrangian takes about 25 h and 30 minutes on 208 cores. This corresponds to an overall speed of about 15000 cell/sec/proc, a remarkably high number when compared to the what was reached on the Fermi architecture (Cineca), that was about 3000 cell/sec/proc on 1024 cores.

Chapter 5

The Calbuco test-case

In this chapter I start presenting some preliminary result obtained using the ASHEE Lagrangian code to simulate the first phase of the Calbuco eruption (Chile 2015, Fig. 1.1 and 5.1). This eruption, categorized as sub-plinian, was characterized by the physical quantity reported in Tab. 5.1.

5.1 The Simulations

Firstly a series of low resolution simulation were run, to determine the input values to be used for the simulation. The plume maximum height is the parameter that has been used to compute the other variables initial values: from precedent studies emerges that this is a function both of temperature and ϕ_{mass}. While the first is rather well constrained for this kind of magma, the second is defined as:

$$\dot{m} = \rho \boldsymbol{v} * A$$

At the end of the conduit $P_{\text{magma}} = P_{\text{atmo}}$ is assumed: in this way both $\boldsymbol{v}_{\text{exit}}$, and ρ are constrained, the first to be equal to $\boldsymbol{v}_{\text{sound}}$ in the mixture, while the second can be computed from the mixture state equation. At this point all initial values were fixed, except for the ϕ_{mass}: a total number of 16 cases was run, varying its value, and keeping all the parameter constant but the volcano vent diameter. The actual

Calbuco, 22th April 2015 - Phase 1	
Duration	5400 s
Maximum plume height	22 Km above sea level
Volume erupted	0.28 - 0.58 km^3
Estimated average mass eruption rate	$0.7 - 7 * 10^7$ kg/s
Temperature	1200 K
Estimated Water mass fraction	$n = 0.05$

Table 5.1: Physical parameters about the eruption at Calbuco, Chile 2015. The eruption was, actually, divided into two separate phases, one with a duration of about 90 minutes, and the second of about 5 hours; data and simulation presented here addresse the former.

Figure 5.1: During the eruption a strong wind was blowing, billowing a huge column of smoke and ash toward Argentina. A strong fallout of ashes from the umbrella cloud is taking place downwind, where the eruptive column is acting as a shield from the wind.

ϕ_{mass} [Kg/s]	Vent Diameter [m]	ϕ_{mass} [Kg/s]	Vent Diameter [m]
4.30E+06	52.97	2.40E+07	125.145
5.50E+06	59.91	2.80E+07	135.17
7.00E+06	67.59	3.30E+07	146.74
8.90E+06	76.21	3.90E+07	159.53
1.10E+07	84.72	4.50E+07	171.36
1.30E+07	92.10	5.20E+07	184.21
1.60E+07	102.18	6.00E+07	197.87
2.00E+07	114.24	6.80E+07	210.65

Table 5.2: Input parameters of the 16 simulations that were run to determine the correct ϕ_{mass} value. The differences in vent diameter implied also a difference in mesh sizes, that were ranging from $22 * 10^4$ to $44 * 10^4$ cells. Bringing to a final $h_{\max} \simeq 22$ Km, $\phi_{\text{mass}} = 1.1 * 10^7$ Kg/s was chosen to be the right guess.

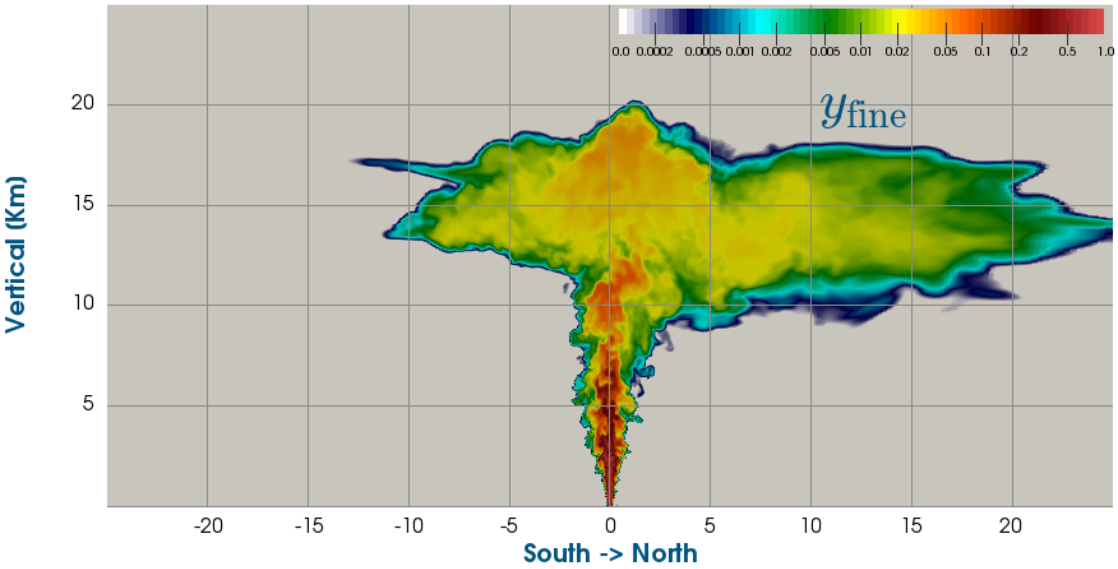

Figure 5.2: Spatial distribution of the fine phase: since this particles are nearly completely coupled with the gas phase, can be safely used as tracers. The settling is almost completely absent for this phase that follows the gas without being able to take part to phenomena as fallout or clustering around large eddies structures.

parameters are reported in Tab. 5.2. This procedure determined that the optimal value for the simulation was $\phi_{\text{mass}} = 1.1 * 10^7$ Kg/s. In future also plume height and umbrella expansion rate, (or the thermal flux), can be used as constraints. By having a larger number of experimental constraints, in facts, it would be possible to explore a more vast range of initial conditions,relaxing, for example the hypothesis that has been made on the atmospheric pressure. It is clear, though, that an increase in experimental constrains would rapidly bring to an amount of initial simulation difficult to manage. This would mean that a more careful technique for sampling of the initial data should also be conceived. After this preparatory tests, a mid resolution simulation, with $N_{\text{cell}} = 3 * 10^6$, was run on 208 cores. The first 2000 seconds of eruption have been reproduced, for a total erupted Lagrangian mass of $2 * 10^7$ Kg, distributed among $2 * 10^6$ Lagrangian parcel, injected at a constant rate of 1000 parcel/sec. In Fig. 5.2- 5.4 the preferential concentration for the three phases: fine, coarse and Lagrangian is reported. From Fig. 5.2 it is evident the almost perfect coupling of the fine phase with the gas: no settling phenomenon is present, i.e. the fall from the umbrella and subsequent deposition on the ground of ashes and particles. Furthermore when looking at the concentration along the z-axis no clustering can be detected, i.e. no zone is present in which the fine phase is being expelled by the large eddy vortex located, with certain periodicity, along the length of the eruptive column: the red color characterizing the center of the plume is a clear indicator of this. In this sense the fine phase can be considered as a tracer, and used to compute the preferential concentration of the other two, according to Eq. 54, Ref. [5].

When looking at Fig. 5.3, instead, both these phenomena can be seen: the 1 mm ash fallout coincides with the red zone located on the right of the volcanic plume. This corresponds, also, to the lee side of the plume: during the eruption, in fact, a

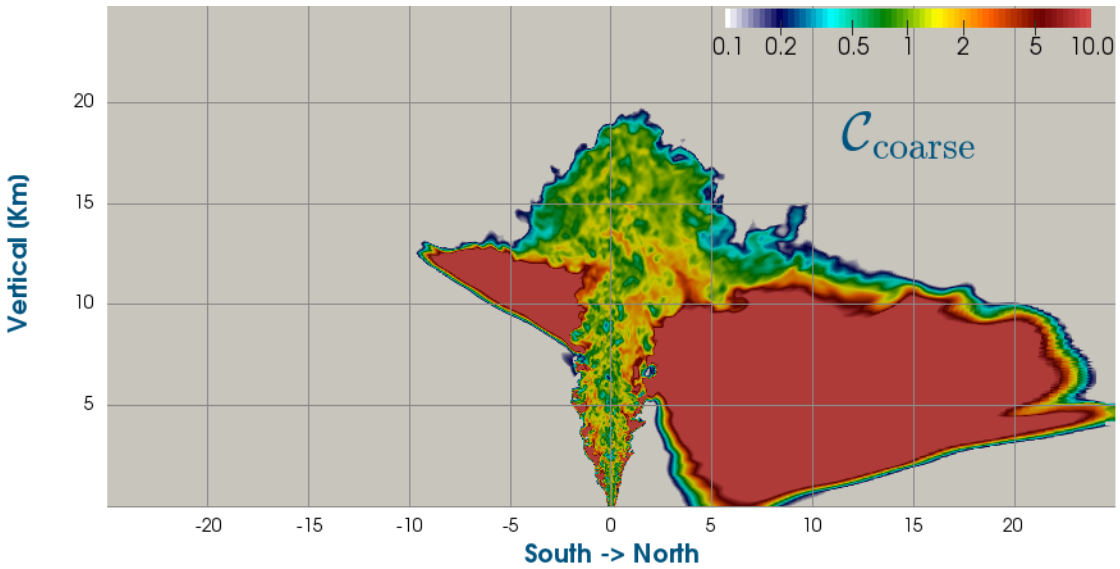

Figure 5.3: Preferential concentration for the coarse phase with respect to y_{fine}: the large, red-colored part, identifies that zone in which, shaded from the wind by the volcanic plume, the settling of the coarse phase takes place. Along the z-axis several, periodic, blue spot can be identified: these correspond to the places where particles are being expelled from the large eddies, highlighting that the Equilibrium-Eulerian model is effective in this region.

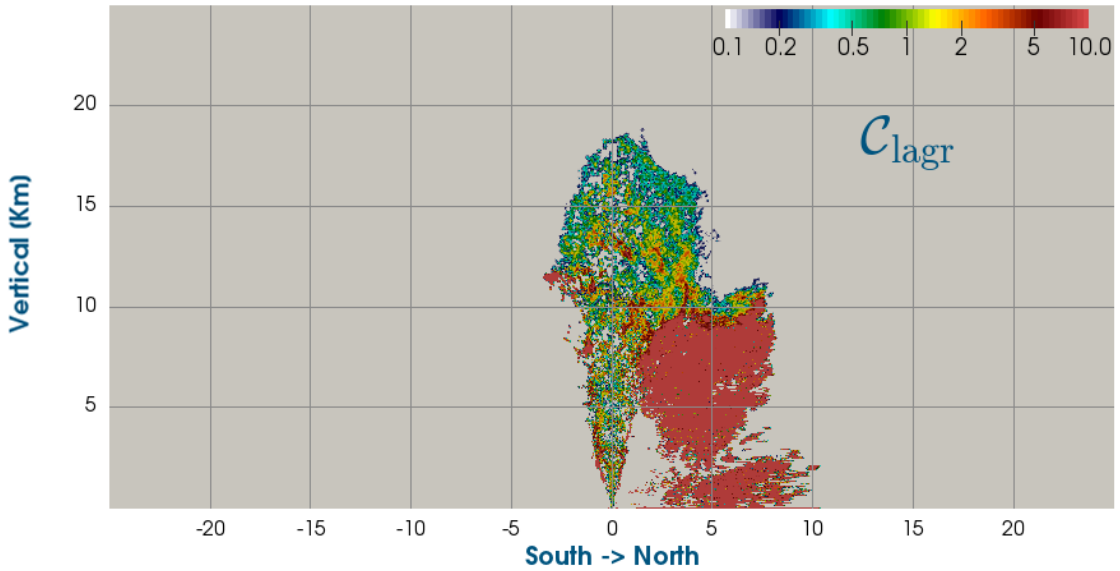

Figure 5.4: Preferential concentration of the Lagrangian particles with respect to the fine phase: zone colored in green marks that area where the Lagrangian phase is acting like a tracer, while red-colored zones mark the presence of clusters: here the decoupling from the fine phase is nearly complete, and, is compared to Fig. 5.3, settling is taking place remarkably closer to the volcano vent. The clustering zones around the large eddies structures are now even more defined.

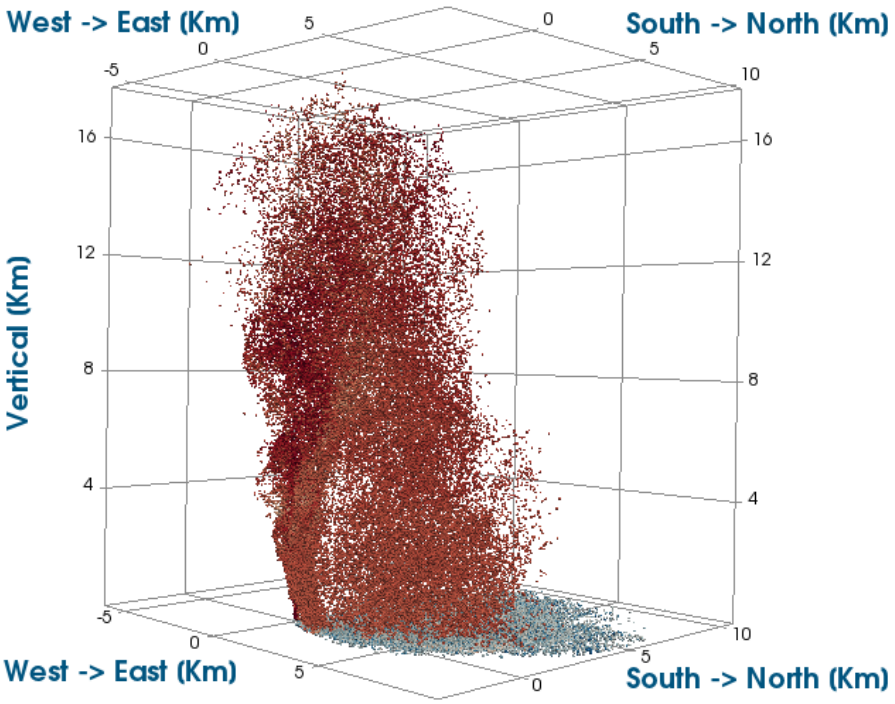

Figure 5.5: 3D visualization of Lagrangian particles, with colors based on their velocity. Wind was blowing North-East and indeed a deposit on the ground is formed on the lee side of the plume (blue particles). The eruptive column, in facts, is acting as a shield from the gusts in this direction.

strong wind was blowing, pushing the column of ash toward Argentina. Shielded from the wind by the plume, ashes and particles fall from the cloud spreading North-East, and deposit on the terrain. Concentrated along the z-axis periodic blue spots can be detected: these correspond to zones in which the coarse phase is at a concentration 10 times smaller with respect to the tracer, i.e. exactly where the eddies are expelling particles. This confirms that the Equilibrium - Eulerian model is adequate to describe this phenomenon [5]. Finally looking at Fig. 5.4 the difference with the coarse phase can be appreciated: even if the displayed patterns are similar, and clustering happens to be in the same places, now the 8 mm ash fallout is more close to the volcano vent (evident also from Fig. 5.5), well reproducing the fact that bigger particles are not able to stay aloft for long times, thus falling in the proximity of the volcano crater (5-10 Km). The clustering around the large eddies is more clear, and the zones in which the concentration is an order of magnitude smaller with respect to the fine phase are now wider along z-axis.

Finally, a comparison between Fig. 5.2 and Fig. 5.1 tells that the simulation reproduces qualitatively well the eruptive event: the spreading of the umbrella cloud, due to the strong wind, is rather neat and accurate. From Fig. 5.1 can be seen also the fallout phenomenon reproduced in Fig.5.3- 5.5. From a more quantitative point fo view, what can be said is that $N_{\text{cells}} = 3 * 10^6$ is the minimum mesh size to have reliable results, i.e. to be able to describe accurately turbulence, or to reproduce clustering patterns [5]. To simulate 2000 s of eruption with this resolution, on 208 cores, a total amount of time of 63 hours was needed. This is (partially) due to the number of Lagrangian parcels, that, when overcoming

the threshold of $N_{\text{parcel}} = 10^6$, significantly slows down the simulation speed: a remarkable difference between the initial and the final number of cell/sec/core was, in fact, measured.

Chapter 6

Conclusions

When started, this work had the ambitious aim to enable the use of simulations of volcanic plumes, by means of the ASHEE code, as an early warning tool at active volcanic site. This requires that a number N~10 of scenarios are run in less than 12 h or 24 h (depending on the choice of the forecasting approach). The planned road-map comprehended, as a first step, the measurement of the computational performance of this code, and, if possible, the implementation of a strategy to improve the efficiency of these simulations. The integration of a Lagrangian library, to enhance the Equilibrium-Eulerian approach of the code, was under study. Since this would have probably brought to some performance lowering, this aspect also needed attention.

After six months of work a precise estimate of execution times can be given, and a best practice to run these simulations has been proposed. This required the study of optimization strategies, both from hardware and software points of view. Whether or not to perform binding to cores, along with the optimal number of cores to use per node, has been determined. The usage of the `renumberMesh` routine, implemented in OpenFOAM, has shown to be rather beneficial to the simulations efficiency. The description of the underlying physics has been improved, with the integration of an OpenFOAM Lagrangian library, the `kinematicCollidingCloud`, that enabled the simulation of particles bigger than 1 mm. This modification has shown to affect performance in an acceptable way, up to a number $N_{\text{parcel}} = 10^6$. As a benchmark for this new version of ASHEE, the eruption occurred in Chile on April 2015, at Calbuco site, has been used. The description of the Lagrangian phase has improved both the description of the proximal fallout, and that of turbulence patterns along the plume z-axis (large-eddies vortex). Present and past studies suggest that a mid resolution (16 cells in the volcano vent diameter, for a total of $1.5 * 10^6$ cells) is good enough to reproduce correctly plume height, release height and rate. However, even when using the optimal cells-to-core ratio (that is around 15000), the simulation time remains above the 25 h. Increasing the number of cores up to 512 (hence decreasing the cells/core ratio, with an overall efficiency loss of about 25%), brings to an execution time of 15 h, thus enabling the forecasting procedure. The simultaneous run of N scenarios can be run in parallel on large supercomputers.

From a more general point of view: this work can be inserted in a rapidly evolving scenario, since a considerable number of research groups is interested in

similar topics and is taking efforts to reach the same goals. Recent pioneering works can be found, that try to implement hybrid parallelization strategies, mixing OpenMPI and OpenMP approaches[17], or exploiting GPUs massively parallel computing. I discarded the idea to test one of these strategies during this thesis, because it was unfeasible within six months, but I think that, if it were possible, joining the forces with one of this groups could bring both exiting collaborations and useful results. Finally I think that great expectation can be pinned on going to new supercomputing architectures such as MARCONI (Cineca infrastructure), since already in moving from FERMI (Cineca) to Laki cluster (INGV), the gain in computing speed per core has been remarkable, paving the way to interesting future developments, in prevision of an actual use of ASHEE for the forecasting of volcanic plumes.

Acknowledgment

The research reported in this work was supported by OGS and CINECA through the HPC-TRES program award number 2015-03 (Stella Valentina Paronuzzi Ticco) and 2015-11 (Matteo Cerminara). Finally, I would like also to mention the useful support given by Giorgio Amati and Ivan Spisso (CINECA), fundamental in the installation and usage of professional tools for the monitoring and analysis of performance. During this work also resources form the ISCRAC project “IscrC Forevolc” where used, on CINECA cluster Galileo. A extended report on “Parallel performance of the new INGV-PI Laki cluster measured with HPL, HPCG, OpenFOAM and ASHEE” has also been published on Rapporti Tecnici INGV, ISSN 2039-7941, 360 (2016).

Bibliography

[1] Barsotti, S., Nannipieri, L., Neri, A.: "MAFALDA: An early warning modeling tool to forecast volcanic ash dispersal and deposition ". Geochem. Geophys. Geosyst., Vol. 9, Is.12, (2008).

[2] Scollo, S., Prestifilippo, M., Spata, G., D'Agostino, M., and Coltelli, M.: "Monitoring and forecasting Etna volcanic plumes" Nat. Hazards Earth Syst. Sci., 9, 1573-1585, (2009).

[3] Cerminara, M., Esposti Ongaro, T., and Berselli, L. C.: "ASHEE-1.0: a compressible, equilibrium-Eulerian model for volcanic ash plumes". Geosci. Model Dev., 9, 697-730 (2016).

[4] Suzuki, Y. J. and Koyaguchi, T.: "Numerical determination of the efficiency of entrainment in volcanic eruption columns". Geophys. Res. Lett., 37, L05302 (2010).

[5] Cerminara, M., Esposti Ongaro, T., and Neri, A. "Large eddy simulation of gas particle kinematic decoupling and turbulent entrainment in volcanic plumes" J. of Vol. and Geot. Res. 326, 143-171, (2016)

[6] Stein, A. F., Draxler, R. R., Rolph, G. D., Stunder, B. J. B., Cohen, M. D., Ngan, F. "NOAA's HYSPLIT Atmospheric Transport and Dispersion Modeling System". Bull. Amer. Meteor. Soc., 96, 2059-2077, (2016)

[7] http://www.cineca.it/fermi-bgq

[8] Valentine, G.A., Wohletz, K.H., "Numerical models of Plinian eruption columns and pyroclastic flows." J. Geophys. Res. 94, 1867-1887 (1989).

[9] Esposti Ongaro, T., Neri, A., Menconi, G., de' Michieli Vitturi, M., Marianelli, P., Cavazzoni, C., Erbacci, G., and Baxter, P. J.: "Transient 3D numerical simulations of column collapse and pyroclastic density current scenarios at Vesuvius". J. Volcanol. Geotherm. Res., 178, 378-396, (2008).

[10] Ferry, J. and Balachandar, S. "A fast Eulerian method for disperse two-phase flow". Int. J. Multiph. Flow, 27, 1199-1226, (2001).

[11] ipm-hpc.sourceforge.net

[12] http://icl.cs.utk.edu/papi/

[13] http://icl.cs.utk.edu

[14] VMSG

[15] VHub

[16] Wen, C. Y. and Yu, Y. H., "Mechanics of Fluidization". Chem. Eng. Prog. Symp. Ser. 62, pp. 100-111, (1966).

[17] Dagnaa, P., Hertzerb, J. "Evaluation of Multi-threaded OpenFOAM Hybridization for Massively Parallel Architectures". Available online at www.praceri.eu